ELEC / COMP 177 – Fall 2011

Computer Networking → Email (SMTP, POP, IMAP) Domain Name System (DNS)

Some slides from Kurose and Ross, *Computer Networking*, 5th Edition

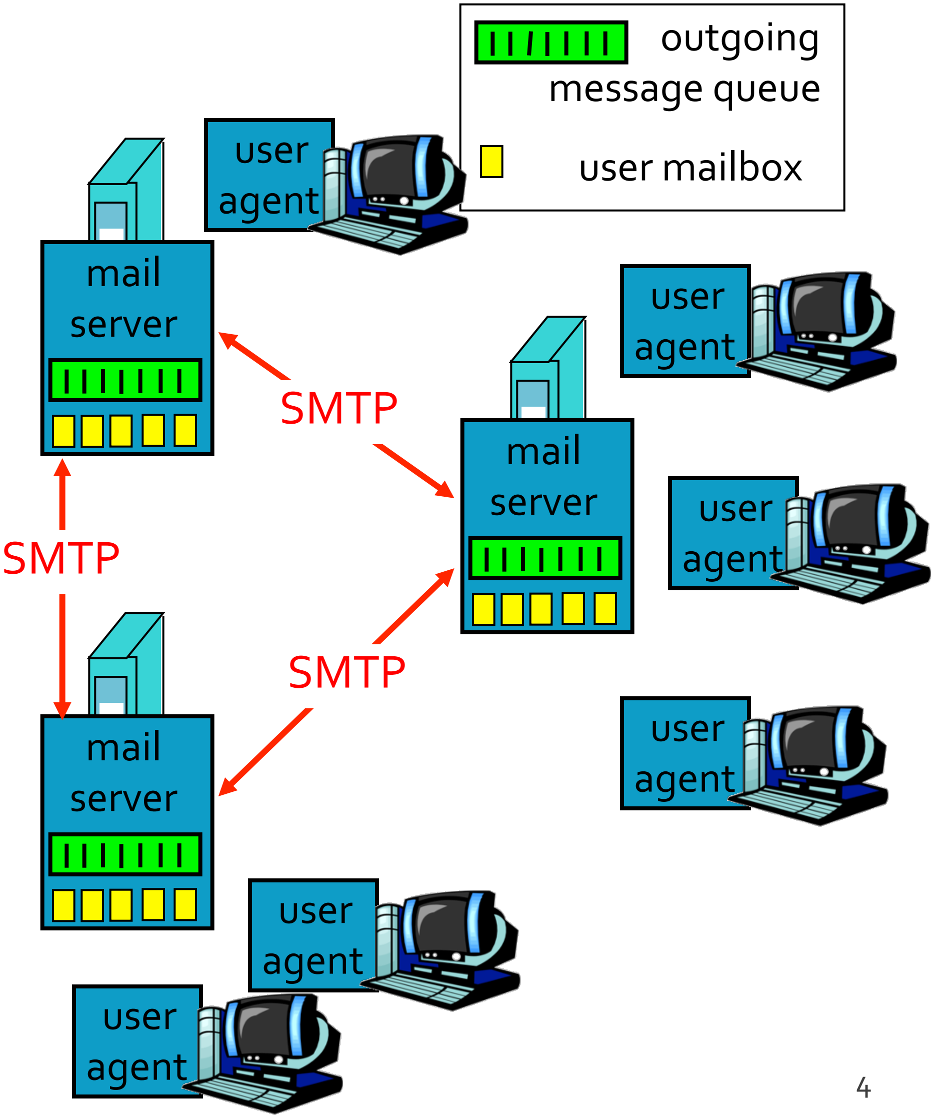
Upcoming Schedule

- **Homework #1**
  - Due Thursday
  - Submit PDF file online via Sakai
  - **Questions?**
    - Office Hours Tue 1:30-3pm, Wed 1:30-3pm

Email (SMTP, POP, IMAP)

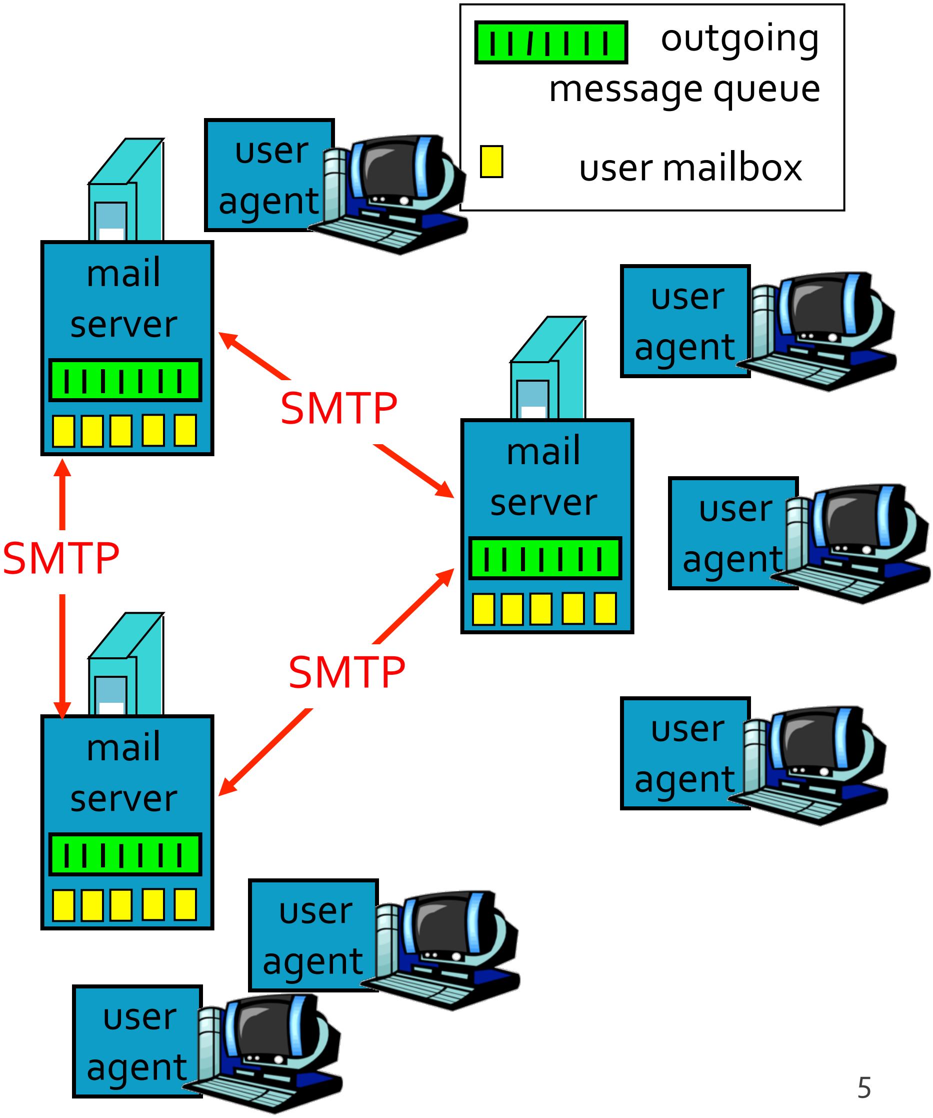
Electronic Mail

- **Three major components**
  - User agents
  - Mail servers
  - Protocol for message transfer (SMTP)
- **User Agent**
  - Your mail reader
    - Composing, editing, reading mail messages
    - e.g., Outlook, Thunderbird, Mail (Mac), …
  - Outgoing and incoming messages stored on server



Electronic Mail – Mail Servers

- **Mail Servers**
  - **Mailbox** contains incoming messages for user
  - **Message queue** of outgoing mail messages (to be sent)
- **SMTP protocol**
  - Used to move email messages between mail servers
  - Client: sending mail server
  - Server: receives messages



Simple Mail Transport Protocol (SMTP)

- Uses TCP to reliably transfer email message from client to server, port 25
- Direct transfer: sending server to receiving server
- Three phases of transfer
  - Handshaking (greeting)
  - Transfer of messages
  - Closure
- Command/response interaction
  - **Commands**: ASCII text
  - **Response**: status code and phrase
- Messages must be in 7-bit ASCII
  - Binary attachments are Base64 *encoded*

SMTP Scenario: Alice sends message to Bob

1) Alice uses UA to compose message to `bob@bigschool.edu`

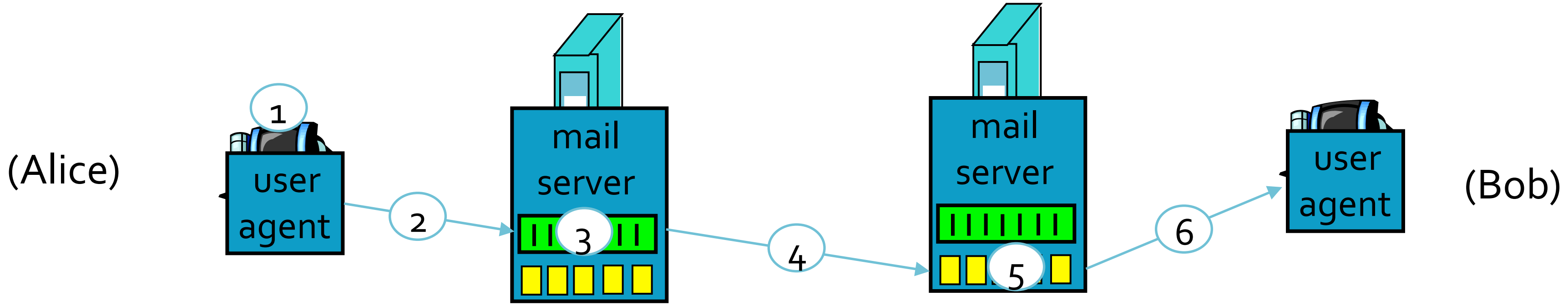
2) Alice's UA sends message to her mail server; message placed in message queue

3) Client side of SMTP opens TCP connection with Bob's mail server

4) SMTP client sends Alice's message over the TCP connection

5) Bob's mail server places the message in Bob's mailbox

6) Bob invokes his user agent to read message



Sample SMTP interaction

S=Server, C=Client

```
S: 220 bigschool.edu
C: HELO smallschool.edu
S: 250  Hello smallschool.edu, pleased to meet you
C: MAIL FROM: <alice@smallschool.edu>
S: 250 alice@smallschool.edu... Sender ok
C: RCPT TO: <bob@bigschool.edu>
S: 250 bob@bigschool.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: This is a test message
C: This is still a test message
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 bigschool.edu closing connection
```

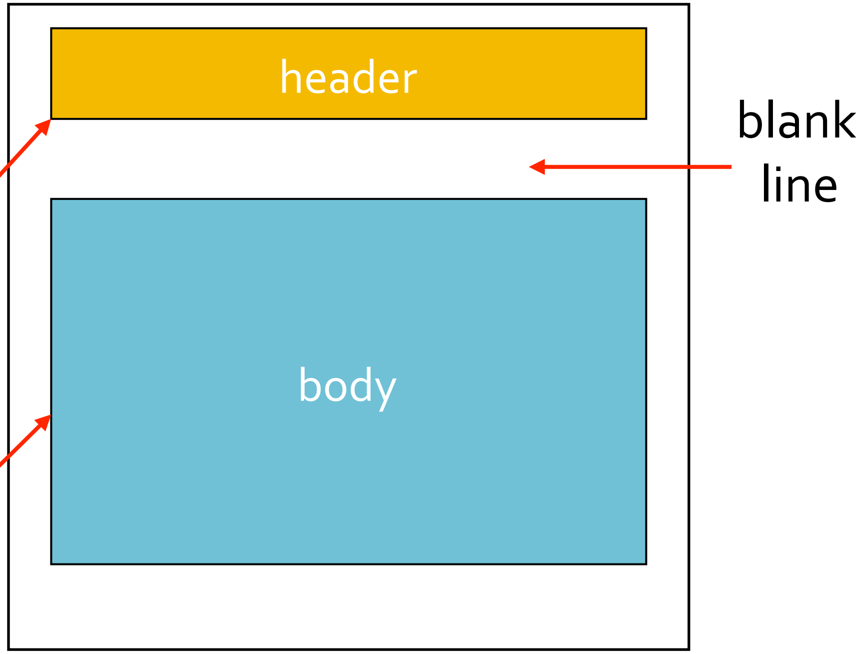
SMTP server uses CRLF.CRLF to determine end of message

SMTP versus HTTP

- “Direction” of transfer
  - HTTP: pull from server
  - SMTP: push to server
- Protocol “style”
  - Both have ASCII command/response interaction and status codes
- Granularity
  - HTTP: each object encapsulated in its own response message *(version 1.0 only)*
  - SMTP: multiple objects sent in multipart message

Mail Message format

- SMTP defines exchanging messages between systems *(transport)*
  - It does **not** specify the format for data inside the message! *(content)*
- RFC 822 defines a standard for text message format
- Header lines
  - To / From / Subject / …
  - **Different from SMTP commands!**
- Body
  - The "message", ASCII characters only

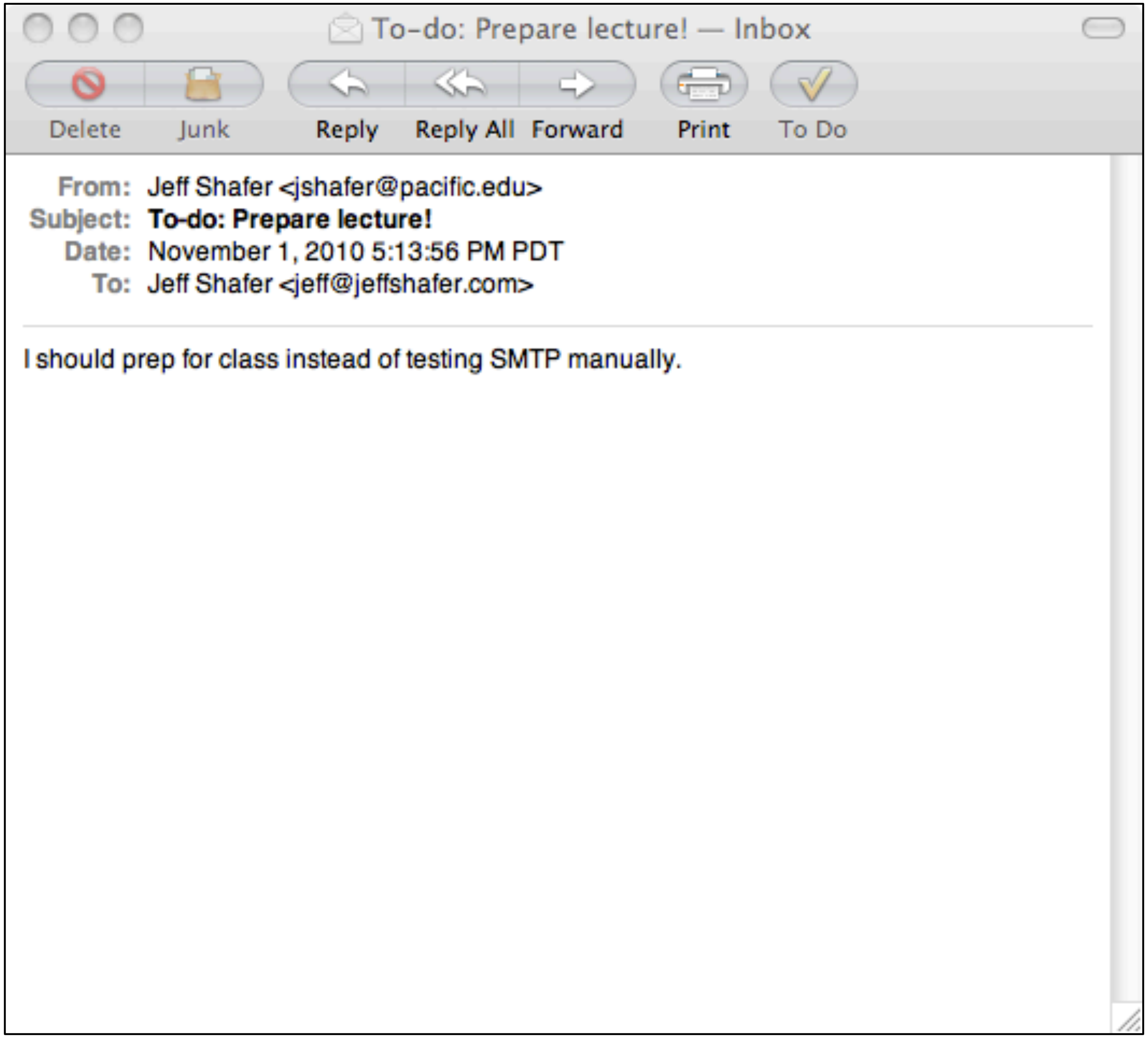


SMTP Manually

```
MacBookPro-Pacific:~ shafer$ telnet smtp.pacific.edu 25
Trying 192.168.100.100...
Connected to smtp.pacific.edu.
Escape character is '^]'.
220 smtp.pacific.edu ESMTP
HELO pacific.edu
250 smtp.pacific.edu
MAIL FROM: <jshafer@pacific.edu>
250 2.1.0 Ok
RCPT TO: <jeff@jeffshafer.com>
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
To: "Jeff Shafer" <jeff@jeffshafer.com
From: "Jeff Shafer" <jshafer@pacific.edu>
Subject: To-Do: Prepare lecture!

I should prep for class instead of testing SMTP manually.
.
250 2.0.0 Ok: queued as C107E39808
QUIT
221 2.0.0 Bye
```

SMTP Manually – The result!



From: Jeff Shafer <jshafer@pacific.edu>
Subject: **To-do: Prepare lecture!**
Date: November 1, 2010 5:13:56 PM PDT
To: Jeff Shafer <jeff@jeffshafer.com>

I should prep for class instead of testing SMTP manually.

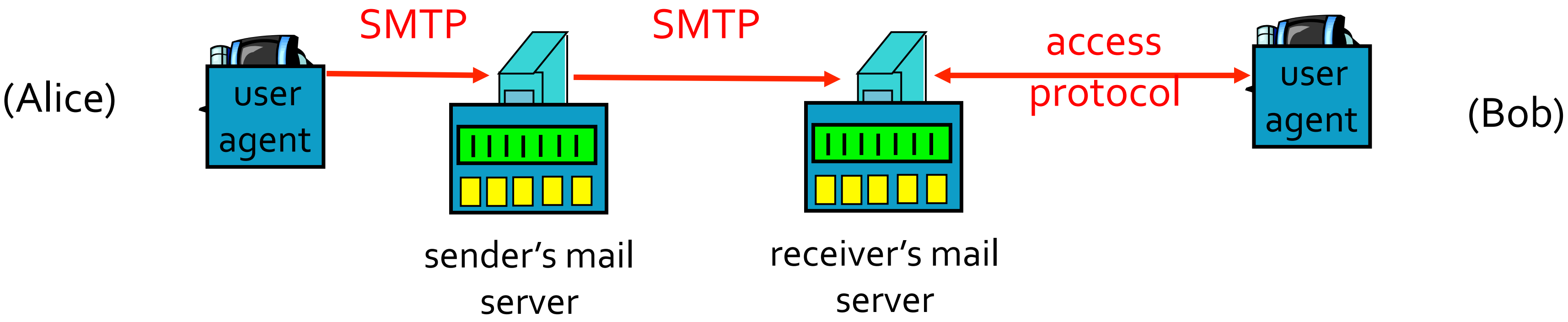
SMTP and SPAM

- Telnet example – did not have to log in!
  - Security an afterthought in original design
- Open relay
  - SMTP server that sends mail to *all* destinations for *all* clients
  - Typically blacklisted today in spam filters
- Optional security measures
  - Only accept clients inside your network?
    - `smtp.pacific.edu` will not respond on port 25 when I'm at home
  - Only accept destinations inside your network?
  - Require users to login? (ESMTP)

SMTP and SPAM

- You can lie to an SMTP server
  - Instead of claiming to be jshafer@pacific.edu, I could have said I was president@pacific.edu
- Countermeasures?
  - `smtp.pacific.edu` could prevent this by forcing me to log on
- What if I send mail via my own SMTP server?
  - Spam filter **challenge**
  - SPF – Sender Protection Framework
    - Puts notes into DNS specifying which IPs are allowed to send mail claiming to be from `pacific.edu`

Mail Access Protocols



- SMTP: delivery/storage to receiver's server
- Mail access protocol: retrieval from server
  - **POP**: Post Office Protocol
    - Authorization (agent <-->server) and download
  - **IMAP**: Internet Mail Access Protocol
    - More features (more complex)
    - Manipulation of stored messages on server
  - **HTTP**: Gmail, Hotmail, Yahoo! Mail, etc.

POP3 Protocol

Authorization phase

- Client commands:
  - **`user`**: declare username
  - **`pass`**: password
- Server responses
  - **`+OK`**
  - **`-ERR`**

Transaction phase, client:

- **`list`**: list message numbers
- **`retr`**: retrieve message by number
- **`dele`**: delete
- **`quit`**

```
S: +OK POP3 server ready
C: user bob
S: +OK
C: pass hungry
S: +OK user successfully logged on
```

```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: <message 1 contents>
S: .
C: dele 1
C: retr 2
S: <message 1 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

POP3

- Previous example uses “download and delete from server” mode.
  - Bob cannot re-read e-mail if he changes client
- “Download and keep on server” mode
  - Allows copies of messages on different clients
- POP3 is stateless across sessions

Internet Message Access Protocol (IMAP)

- Keep all messages in one place: the server
  - Clients might have a temporary *cache* for offline access
- Allows user to organize messages in folders
- IMAP keeps user state across sessions:
  - Names of folders and mappings between message IDs and folder name
- Other features
  - Server-side searches (don’t have to download mailbox!)
  - Multiple concurrent clients
- *Can explore more in your homework assignment*

Domain Name System (DNS)

Motivation

- IP addresses are hard to remember
  - `198.16.253.143`? Or was it `.146`?
- Human-friendly names are much better
  - `engineering.pacific.edu`
- How can we translate between the two?

Early Days (prior to 1983)

- Each computer on the ARPAnet (early Internet) had a single file
  - `hosts.txt` maps all known host names to IP address
- Master list maintained by SRI Network Information Center
  - Email them if your mapping changes
  - New list produced 1-2 times a week
  - All hosts download the new list
- **Problems with this approach?**

Domain Name System (DNS)

- **Distributed database** implemented in hierarchy of many **name servers**
- **Application-layer protocol**
  - Hosts, routers, and name servers communicate to resolve names (address/name translation)
  - Core Internet function, implemented as application-layer protocol
  - Complexity at network's "edge"

DNS

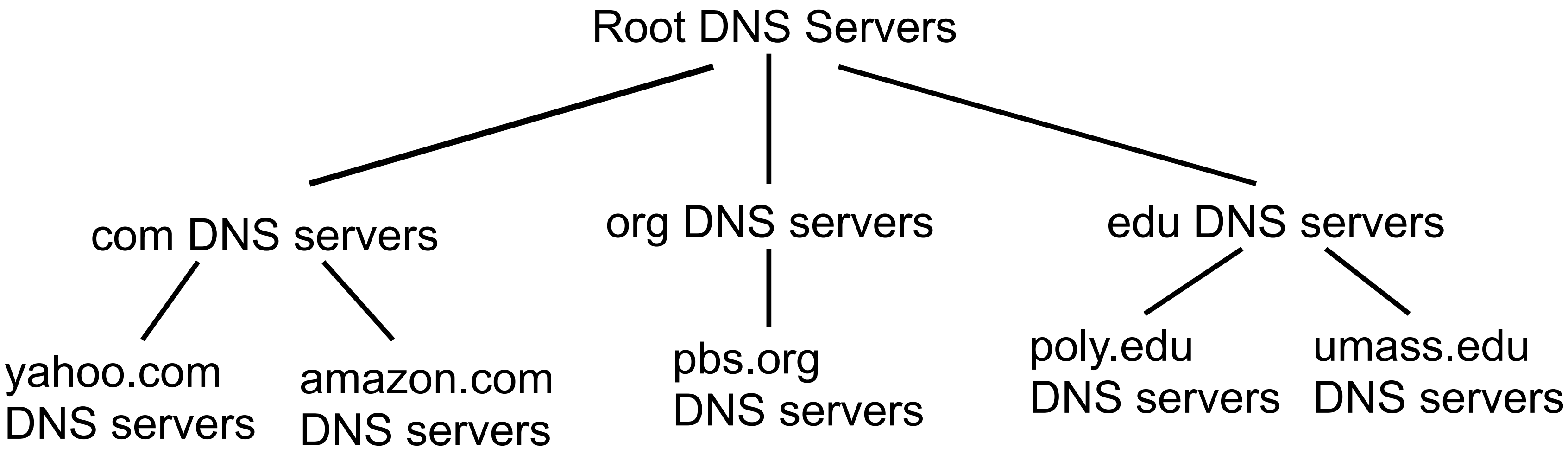
- **DNS services**
  - Hostname to IP address translation
  - Host aliasing
    - Canonical, alias names
  - Mail server aliasing
  - Load distribution
    - Replicated Web servers: set of IP addresses for one canonical name

- Why not centralize DNS?
  - Single point of failure
  - Traffic volume
  - Distant centralized database
  - Maintenance
  - **Doesn’t scale!**

What’s in a Name?

- `engineering.pacific.edu`
  - .edu is top-level domain
  - “pacific” belongs to .edu
  - “engineering” belongs to “pacific”
  - Hierarchical! Read from right to left
- Limits?
  - Up to 127 levels of hierarchy
  - Each label can have up to 63 characters
  - Full domain name cannot exceed 253 characters

Distributed, Hierarchical Database



Root DNS Servers

com DNS servers

org DNS servers

edu DNS servers

yahoo.com DNS servers

amazon.com DNS servers

pbs.org DNS servers

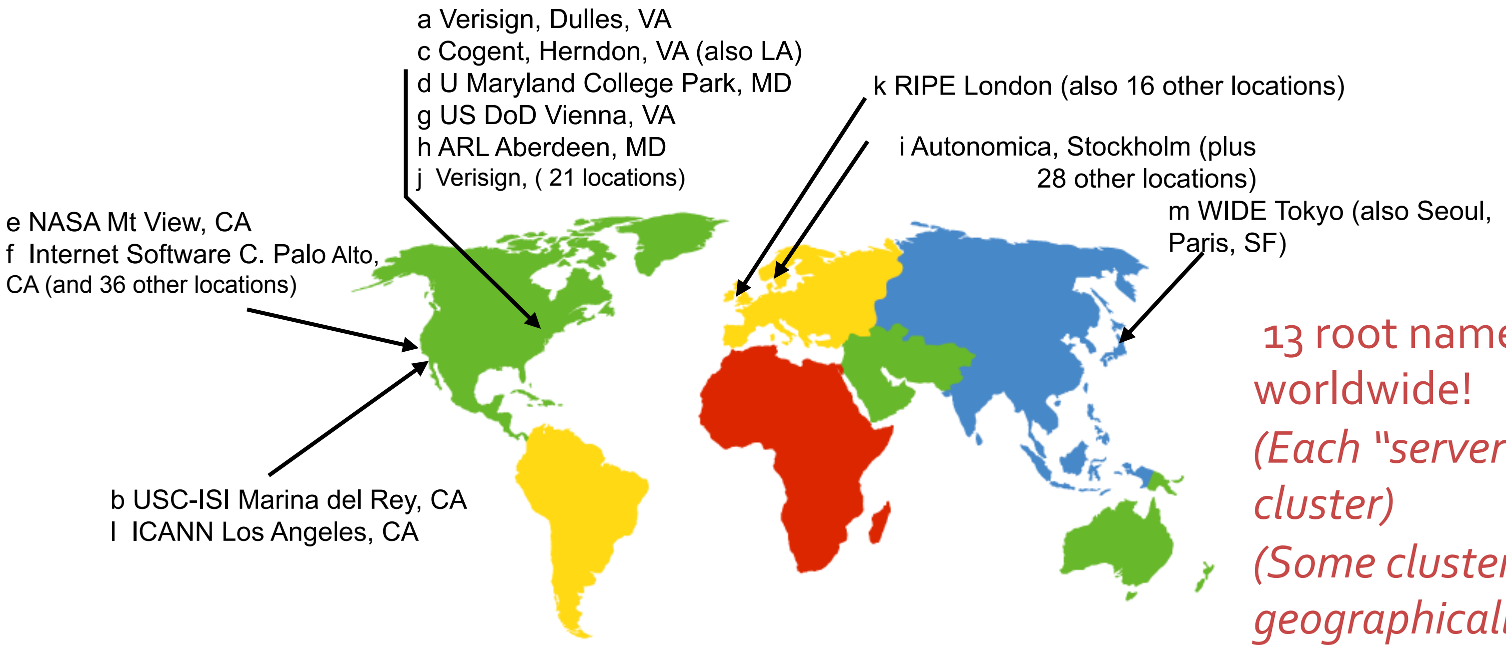
poly.edu DNS servers

umass.edu DNS servers

- Client wants IP for www.amazon.com
  1. Client queries a root server to find com DNS server
  2. Client queries com DNS server to get amazon.com DNS server
  3. Client queries amazon.com DNS server to get IP address for www.amazon.com

DNS: Root name servers

- Contacted by local name server that can not resolve name
- Root name server:
  - Contacts authoritative name server if name mapping not known
  - Gets mapping
  - Returns mapping to local name server



a Verisign, Dulles, VA
c Cogent, Herndon, VA (also LA)
d U Maryland College Park, MD
g US DoD Vienna, VA
h ARL Aberdeen, MD
j Verisign, (21 locations)
k RIPE London (also 16 other locations)
i Autonomica, Stockholm (plus 28 other locations)
m WIDE Tokyo (also Seoul, Paris, SF)
e NASA Mt View, CA
f Internet Software C. Palo Alto, CA (and 36 other locations)
b USC-ISI Marina del Rey, CA
l ICANN Los Angeles, CA

13 root name "servers" worldwide!

(Each "server" is really a cluster)

(Some clusters are geographically distributed)

TLD and Authoritative Servers

- **Top-level domain** (TLD) servers
  - Responsible for com, org, net, edu,… and all top-level country domains (uk, fr, ca, jp, …)
  - Server maintainers
    - VeriSign for com TLD
    - Educause for edu TLD
- **Authoritative** DNS servers:
  - Organization's DNS servers, providing authoritative hostname to IP mappings for organization's servers (e.g., Web, mail).
  - Can be maintained by organization or service provider

Local Name Server

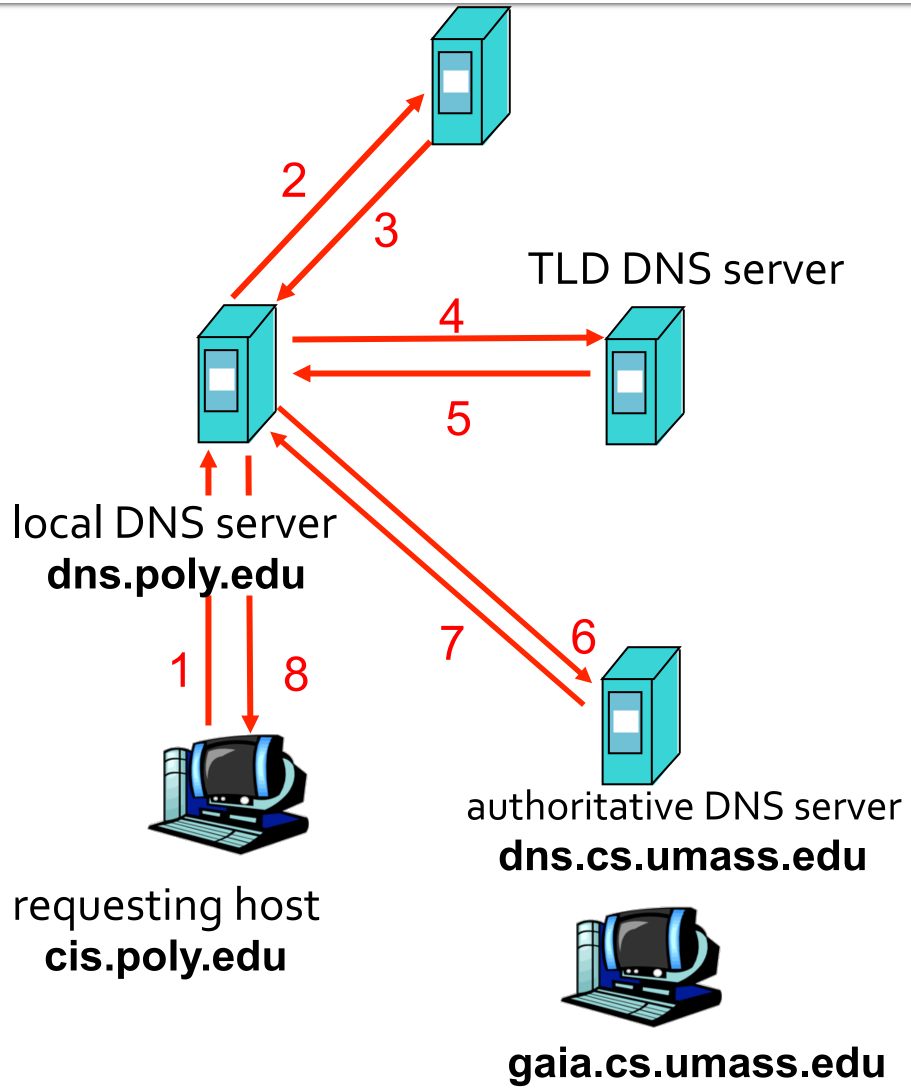
- Does not strictly belong to hierarchy
- Each ISP (residential ISP, company, university) has one.
  - Also called “default name server”
- When host makes DNS query, query is sent to its local DNS server
  - Acts as proxy, forwards query into hierarchy
- **You typically know this server’s IP address from DHCP**

DNS Name Resolution

- Two types
- **Recursive**
  - The server you contact provides the final answer
  - *Behind the scenes, it may make several consecutive requests*
- **Iterative**
  - The server you contact directs you to a different server to get (closer to) the final answer

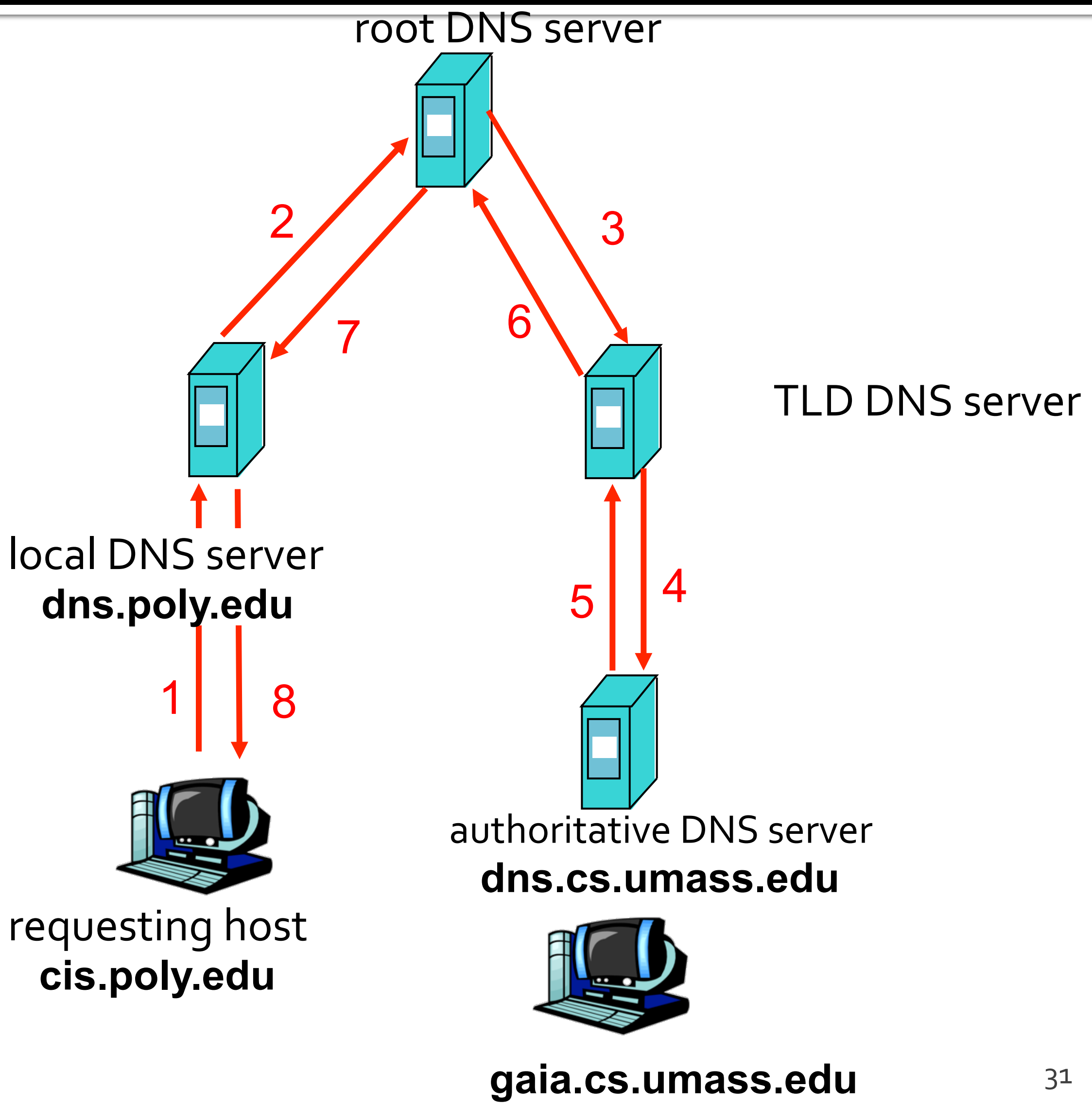
DNS Example

- Host at cis.poly.edu wants IP address for gaia.cs.umass.edu
- Iterative query:
  - Contacted server replies with name of server to contact
  - "I don't know this name, but ask this server"



DNS Example

- Recursive query:
  - Puts burden of name resolution on contacted name server
  - Heavy load?



DNS: Caching and Updating records

- Once (any) name server learns mapping, it **caches** mapping
  - This includes your computer
  - This includes your ISP's name server
- Cache entries eventually timeout
  - Can be specified by the authoritative server, and/or overruled by the local server
- TLD (.com, .net, .org, etc…) servers are typically cached in local name servers
  - Reduces traffic on the root servers!

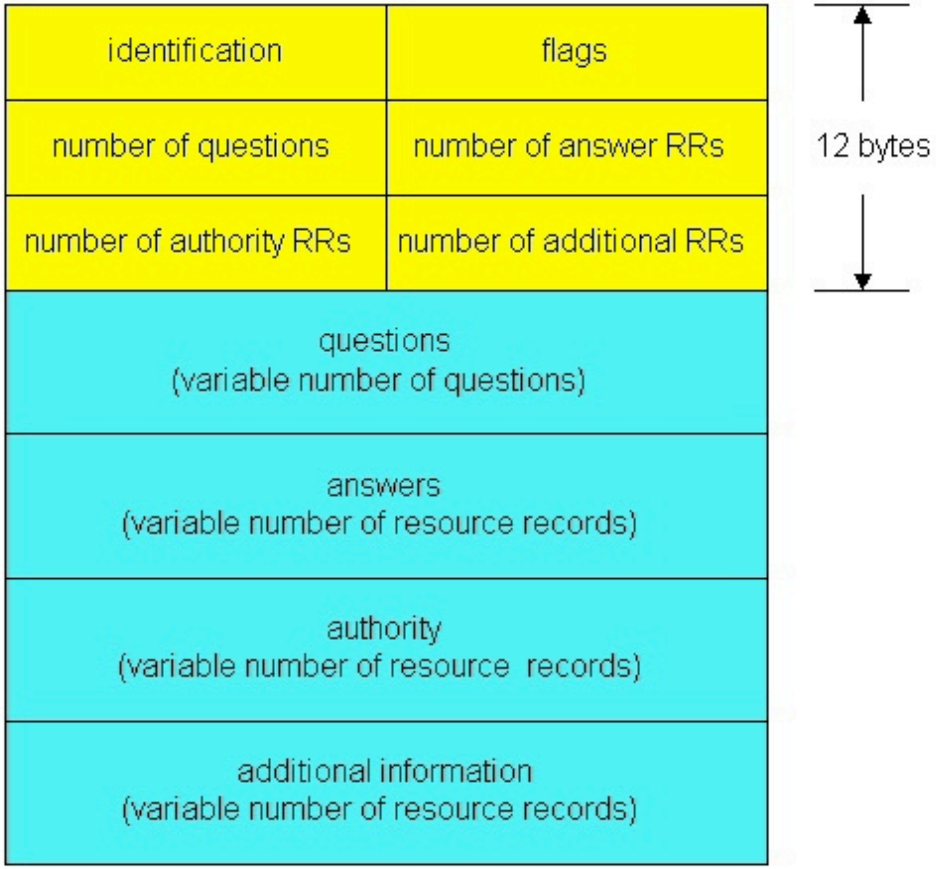
DNS: Distributed DB

Resource Record (RR) format: **`(name, value, type, ttl)`**

- Type=A
  - *name* is **hostname**
  - *value* is **IP address**
- Type=NS
  - *name* is domain (e.g. foo.com)
  - *value* is **hostname** of **authoritative** name server for this domain
- Type=CNAME
  - *name* is alias name for some "canonical" (real) name
  - `www.ibm.com` is really `servereast.backup2.ibm.com`
  - value is canonical name
- Type=MX
  - *value* is name of mailserver associated with name

DNS Protocol

- DNS protocol:
  - Query and reply messages
  - Both use same message format
- Message header
  - Identification: 16 bit # for query, reply to query uses same #
  - Flags:
    - Query or reply
    - Recursion desired
    - Recursion available
    - Reply is authoritative



identification | flags

number of questions | number of answer RRs

number of authority RRs | number of additional RRs

12 bytes

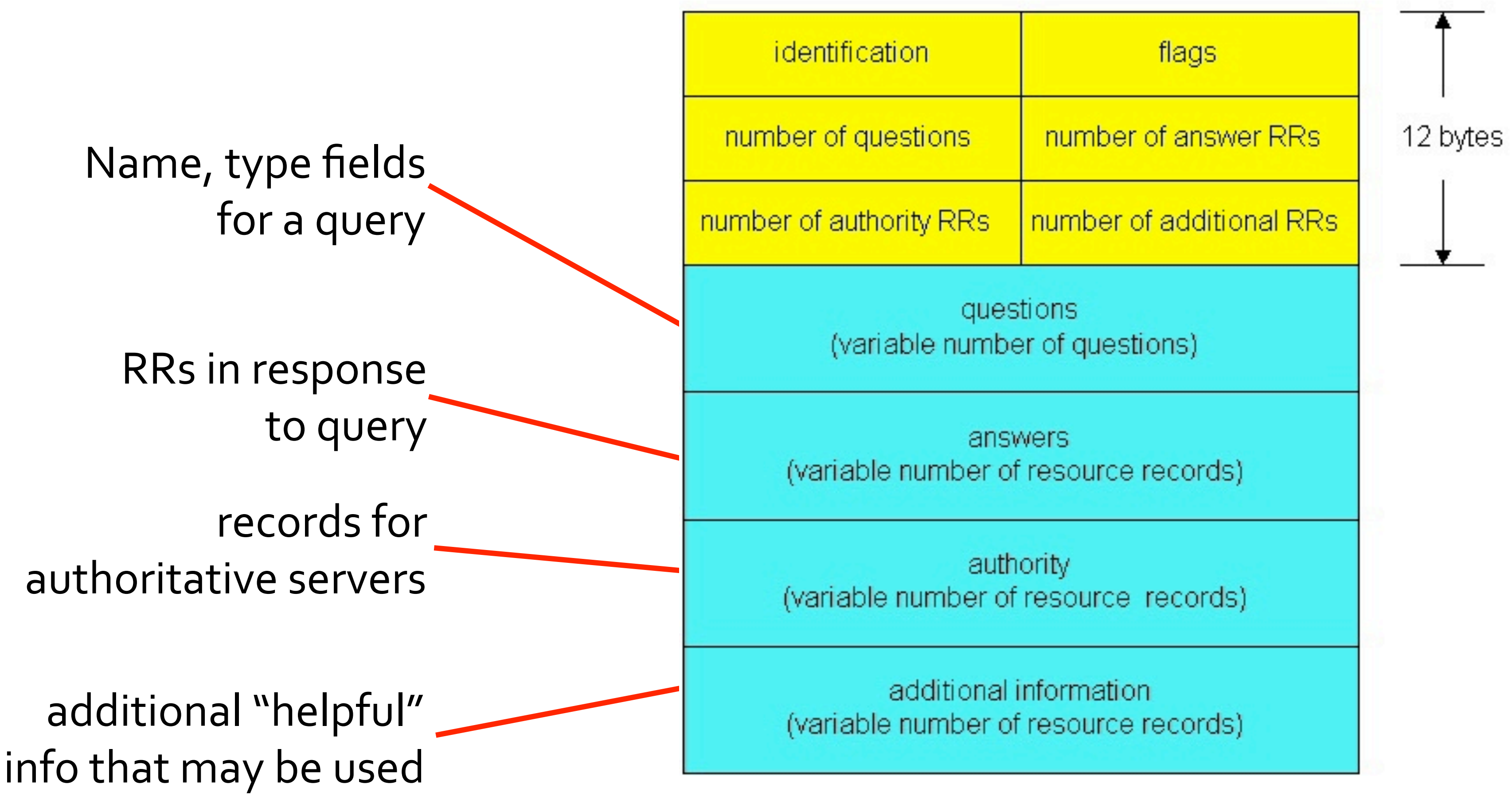
questions
(variable number of questions)

answers
(variable number of resource records)

authority
(variable number of resource records)

additional information
(variable number of resource records)

DNS Protocol



Name, type fields for a query

RRs in response to query

records for authoritative servers

additional "helpful" info that may be used

identification | flags

number of questions | number of answer RRs

number of authority RRs | number of additional RRs

12 bytes

questions (variable number of questions)

answers (variable number of resource records)

authority (variable number of resource records)

additional information (variable number of resource records)

Inserting records into DNS

- Example: creating jeffshafer.com website
- Register name jeffshafer.com at *DNS registrar* (e.g. PairNIC.com, GoDaddy.com, etc…)
  - Provide names and IP addresses of authoritative name server (primary and secondary)
  - Registrar inserts two RRs into com TLD server:

```
(jeffshafer.com, dns1.jeffshafer.com, NS)
(dns1.jeffshafer.com, 212.212.212.1, A)
```

- Authoritative server should contain records too
  - Type A record for jeffshafer.com and/or www.jeffshafer.com
  - Type MX record for jeffshafer.com
- **Where does the IP address of your website and authoritative server come from?**

Multipurpose DNS

- What else do we stuff into DNS records?
  - SPF entries for email
    - Anti-spam
  - MX records for email
    - What are the **multiple** host names that receive mail for this domain?
    - 1st priority, then 2nd backup, then 3rd backup, etc…
    - Why do I need this?
      - jeffshafer.com uses Google App Service for email

DNS and UDP

- DNS uses UDP by default
  - It *can* use TCP, but it's rare
  - **Isn't this unreliable?**
- Why use UDP
  - Faster (in three ways!)
    - No need to establish a connection (RTT/latency overhead)
    - Lower per-packet byte overhead in UDP header
    - Less packet processing by hosts
  - Reliability not needed
    - DNS will just re-request if no response received (2-5 seconds)

Dynamic Host Configuration Protocol (DHCP)

DHCP Overview

- How does a host obtain its IP address?
  - DHCP – Dynamic Host Configuration Protocol
- DHCP is an application
  - But it is interested in IP address information
  - That is part of the network layer! (two layers down!)

Assignment of IP Addresses

- How does a host computer gets its IP address?
- Static assignment
  - Requires **user** involvement to set in OS
  - We configure hosts in the lab statically
    - It’s “educational!” (plus, to make each lab work, you have to be very careful about what IP addresses you use)
  - Datacenters might configure servers statically since they rarely change addresses
- Dynamic assignment
  - Requires no user involvement
  - Represents the bulk of hosts on the Internet

Dynamic Host Configuration Protocol (DHCP)

- Goals of DHCP
  - Plug and play!
    (Can’t trust grandma to set her IP address, netmask, and default gateway correctly…)
  - Allow host to *dynamically* obtain its IP address from network server when it joins network
  - Allow host to renew its lease on in-use address
  - Allow reuse of addresses (if you disconnect your host, someone else can use that address)

DHCP

- DHCP packet nested inside UDP, IP, and Ethernet frame
- Four stages to DHCP
  1. Discover *(new host only)*
  2. Offer *(new host only)*
  3. Request
  4. Acknowledge

Step 1 – DHCP Discover

- "Discover DHCP servers on the network"
- (New host only) Host **broadcasts** "DHCP discover" message to entire subnet
  - **What is broadcast?**
  - Subnet = Anywhere on Ethernet you can reach without going through a router
  - DHCP server either located on same subnet, or router has been configured to intercept and forward DHCP messages
  - Router might **be** the DHCP server!

Step 2 – DHCP Offer

- "DHCP servers offer client an IP assignment"
- (New host only) DHCP server responds directly to client with "DHCP offer" message
- Message contains
  - IP address of DHCP server
  - A lease offer to the client
    - IP address
    - Subnet mask
    - Lease duration
- Might get several offers from different DHCP servers

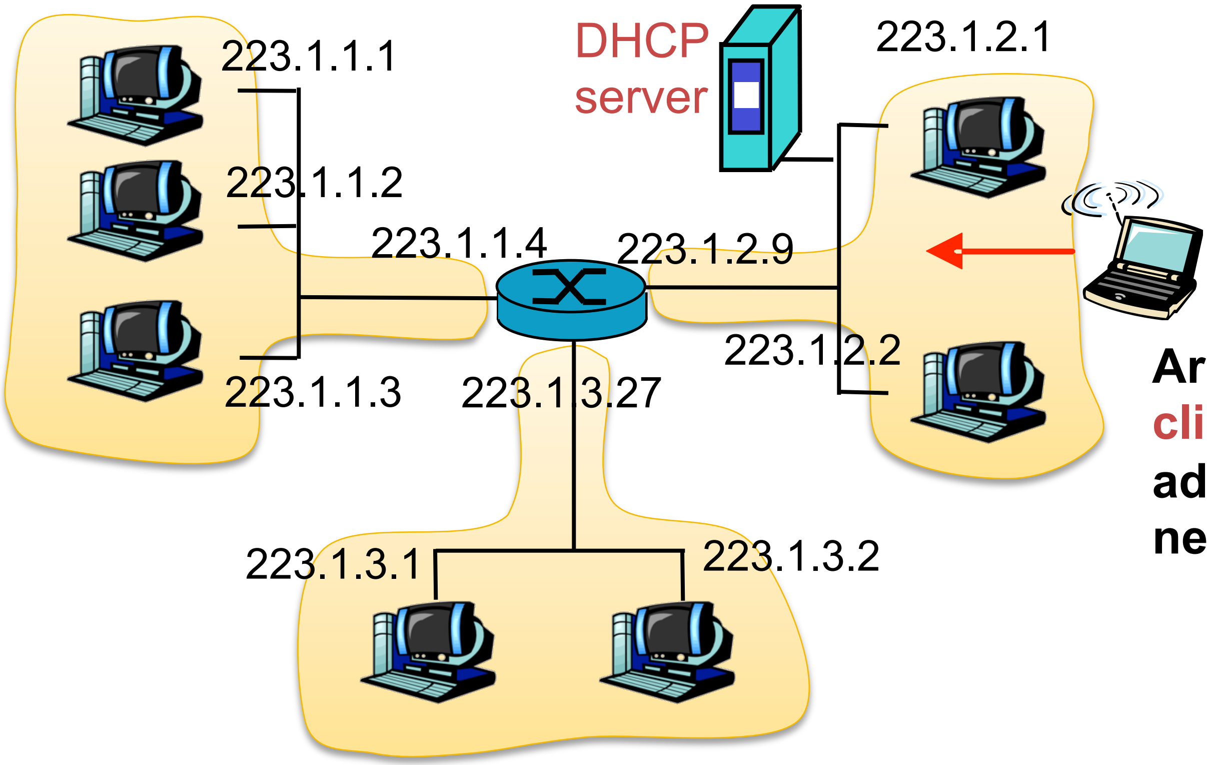
Step 3 – DHCP Request

- "Host requests the best offer"
- Host picks the DHCP offer it likes best
- Host requests IP address with a "DHCP request" message
  - Message is **broadcast across subnet**. **Why?**
    - May have received multiple offers from multiple servers
    - Servers are reserving an IP address for you
    - Need to let all servers know, even the ones you didn't accept (so they can return the address to the pool)

Step 4 – DHCP Ack

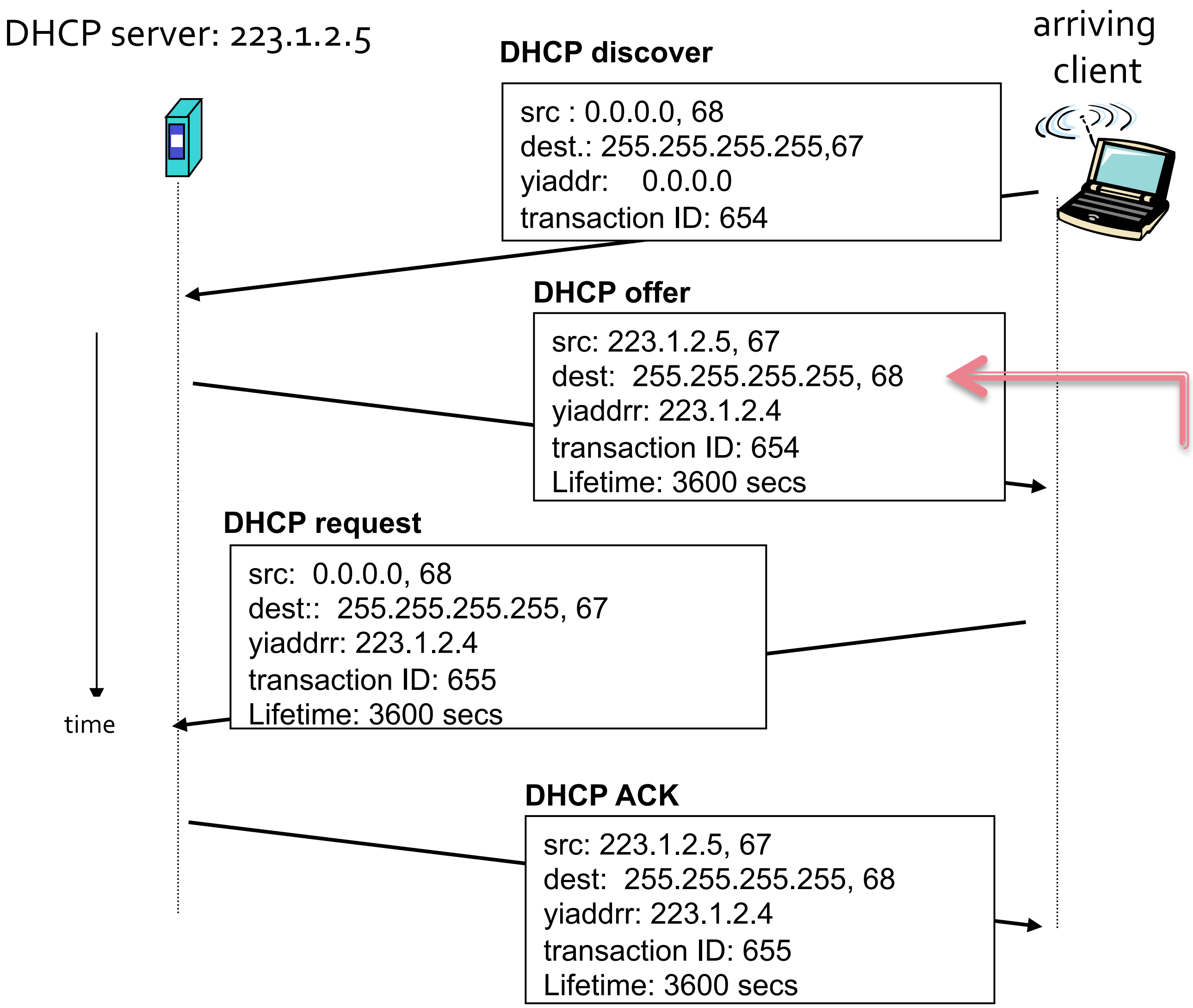
- “DHCP server confirms accepted offer, and sends other information.”
- Only the server whose lease the client requested sends back a “DHCP Ack” message
- Re-confirms the lease information

DHCP Client-Server Scenario



Arriving DHCP client needs address in this network

DHCP client-server scenario



DHCP discover

src : 0.0.0.0, 68
dest.: 255.255.255.255,67
yiaddr: 0.0.0.0
transaction ID: 654

DHCP offer

src: 223.1.2.5, 67
dest: 255.255.255.255, 68
yiaddrr: 223.1.2.4
transaction ID: 654
Lifetime: 3600 secs

DHCP request

src: 0.0.0.0, 68
dest:: 255.255.255.255, 67
yiaddrr: 223.1.2.4
transaction ID: 655
Lifetime: 3600 secs

DHCP ACK

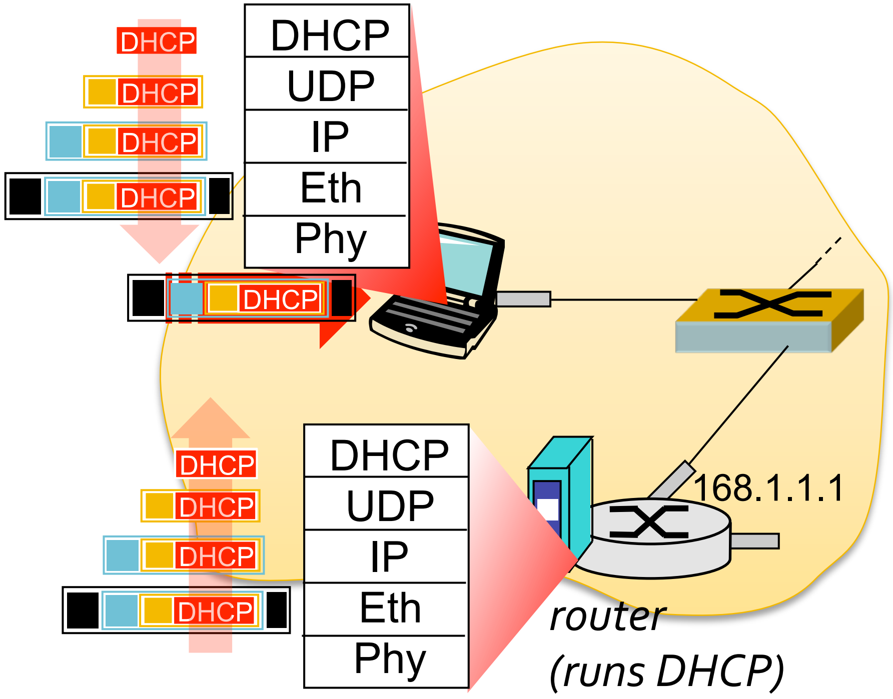
src: 223.1.2.5, 67
dest: 255.255.255.255, 68
yiaddrr: 223.1.2.4
transaction ID: 655
Lifetime: 3600 secs

Why is this broadcast?

DHCP – More Than Just IP Address

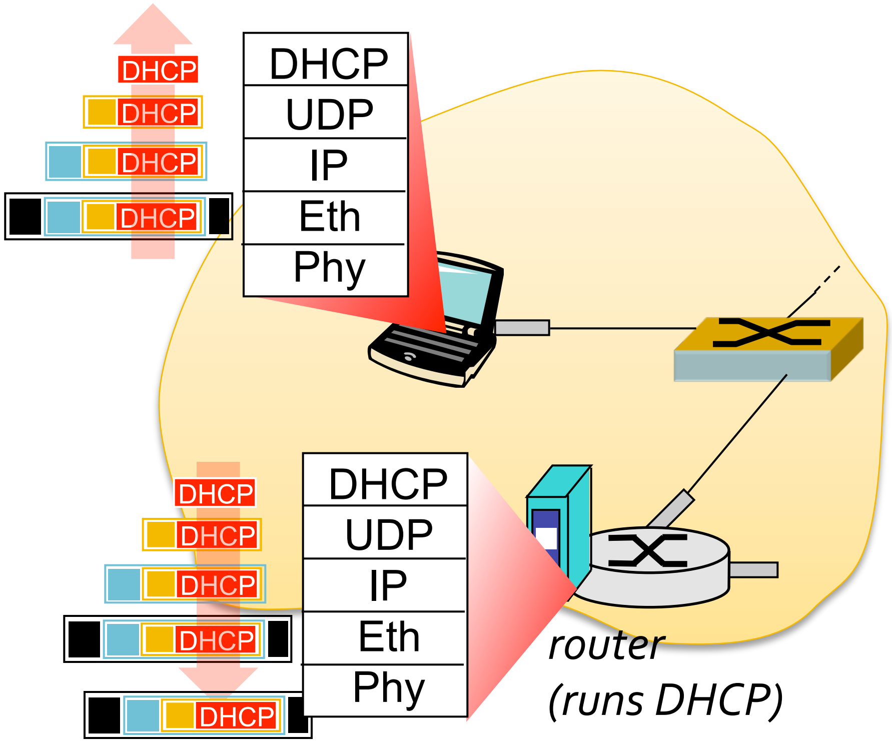
- DHCP can return more than just allocated IP address on subnet
  - Address of gateway router for client
  - Name and IP address of DNS sever(s)
  - Network mask (indicating network versus host portion of address)
  - NTP server (network time)
  - LDAP server (address book)
  - SIP server (Voice-over-IP server)
  - … and many many more possibilities!

DHCP: example



- Connecting laptop needs its IP address, addr of first-hop router, addr of DNS server: use DHCP
- DHCP request encapsulated in UDP, encapsulated in IP, encapsulated in Ethernet
- Ethernet frame broadcast (dest: FFFFFFFFFFFF) on LAN, received at router running DHCP server
- Ethernet unpacked (to IP, then UDP, then DHCP)

DHCP: example



- DCP server formulates DHCP ACK containing client's IP address, IP address of first-hop router for client, name & IP address of DNS server
- encapsulation of DHCP server, frame forwarded to client, demux'ing up to DHCP at client
- client now knows its IP address, name and IP address of DSN server, IP address of its first-hop router

DHCP Wireshark Output @ Pacific

(Shortened, and I already had an old IP)

DHCP REQUEST

```
Ethernet II, Src: 7c:6d:62:8c:c2:df, Dst: Broadcast
      (ff:ff:ff:ff:ff:ff)
IP, Src: 0.0.0.0, Dst: 255.255.255.255
UDP, Src Port: bootpc (68), Dst Port: bootps (67)
Bootstrap Protocol
  Message type: Boot Request (1)
  Hardware type: Ethernet
  Transaction ID: 0x73487c67
  Bootp flags: 0x0000 (Unicast)
  Client IP address: 0.0.0.0 (0.0.0.0)
  Your (client) IP address: 0.0.0.0 (0.0.0.0)
  Next server IP address: 0.0.0.0 (0.0.0.0)
  Relay agent IP address: 0.0.0.0 (0.0.0.0)
  Client MAC address: Apple_8c:c2:df (7c:6d:62:8c:c2:df)
  Magic cookie: DHCP
  Option: (t=53,l=1) DHCP Message Type = DHCP Request
  Option: (t=55,l=10) Parameter Request List
  Option: (t=57,l=2) Maximum DHCP Message Size = 1500
  Option: (t=61,l=7) Client identifier
  Option: (t=50,l=4) Requested IP Address = 10.10.207.20
  Option: (t=51,l=4) IP Address Lease Time = 90 days
  Option: (t=12,l=18) Host Name = "MacBookPro-Pacific”
```

DHCP ACK

```
Ethernet II, Src: Cisco_53:3f:fc (00:05:dc:53:3f:fc), Dst: 7c:6d:
      62:8c:c2:df
IP, Src: 10.10.207.254, Dst: 10.10.207.20
UDP, Src Port: bootps (67), Dst Port: bootpc (68)
Bootstrap Protocol
  Message type: Boot Reply (2)
  Hardware type: Ethernet
  Transaction ID: 0x73487c67
  Bootp flags: 0x0000 (Unicast)
  Client IP address: 0.0.0.0 (0.0.0.0)
  Your (client) IP address: 10.10.207.20 (10.10.207.20)
  Next server IP address: 0.0.0.0 (0.0.0.0)
  Relay agent IP address: 10.10.207.254 (10.10.207.254)
  Client MAC address: Apple_8c:c2:df (7c:6d:62:8c:c2:df)
  Magic cookie: DHCP
  Option: (t=53,l=1) DHCP Message Type = DHCP ACK
  Option: (t=54,l=4) DHCP Server Identifier = 10.10.4.226
  Option: (t=51,l=4) IP Address Lease Time = 1 day
  Option: (t=1,l=4) Subnet Mask = 255.255.254.0
  Option: (t=3,l=4) Router = 10.10.207.254
  Option: (t=6,l=8) DNS= 10.10.4.2.226, 10.10.4.227
  Option: (t=15,l=15) Domain Name = "eng.pacific.edu"
  Option: (t=44,l=8) NetBIOS over TCP/IP Name Server
  Option: (t=46,l=1) NetBIOS over TCP/IP Node Type = H-node
```

How to Allocate Addresses?

- DHCP server has a pool of addresses
  - How to we give them out to clients?
- Randomly?
  - First-come, first-serve
  - Host might get a different address each time
- Persistently?
  - Look at host MAC address, and try to give it the same address it had last time
- Statically?
  - Reserve an IP address only for a specific client with a specific MAC address

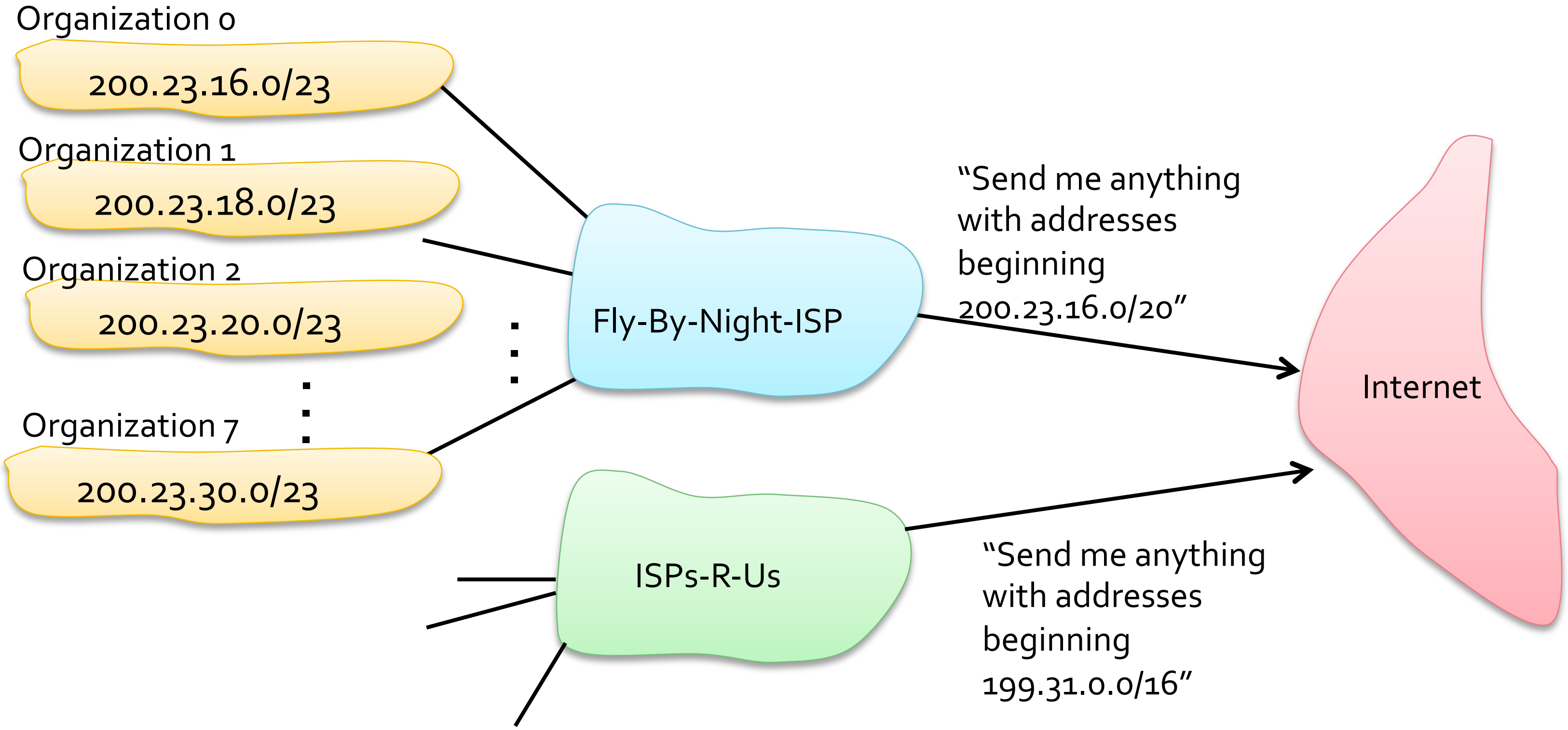
Source of IP Addresses

- How do the network DHCP servers know what pool of IP addresses to use?
  - One way: Your network is allocated portion of its provider ISP's address space

ISP's block	11001000 00010111 00010000 00000000	(200.23.16.0/20)
Organization 0	11001000 00010111 00010000 00000000	(200.23.16.0/23)
Organization 1	11001000 00010111 00010010 00000000	(200.23.18.0/23)
Organization 2	11001000 00010111 00010100 00000000	(200.23.20.0/23)
...	….. ….	….
Organization 7	11001000 00010111 00011110 00000000	(200.23.30.0/23)

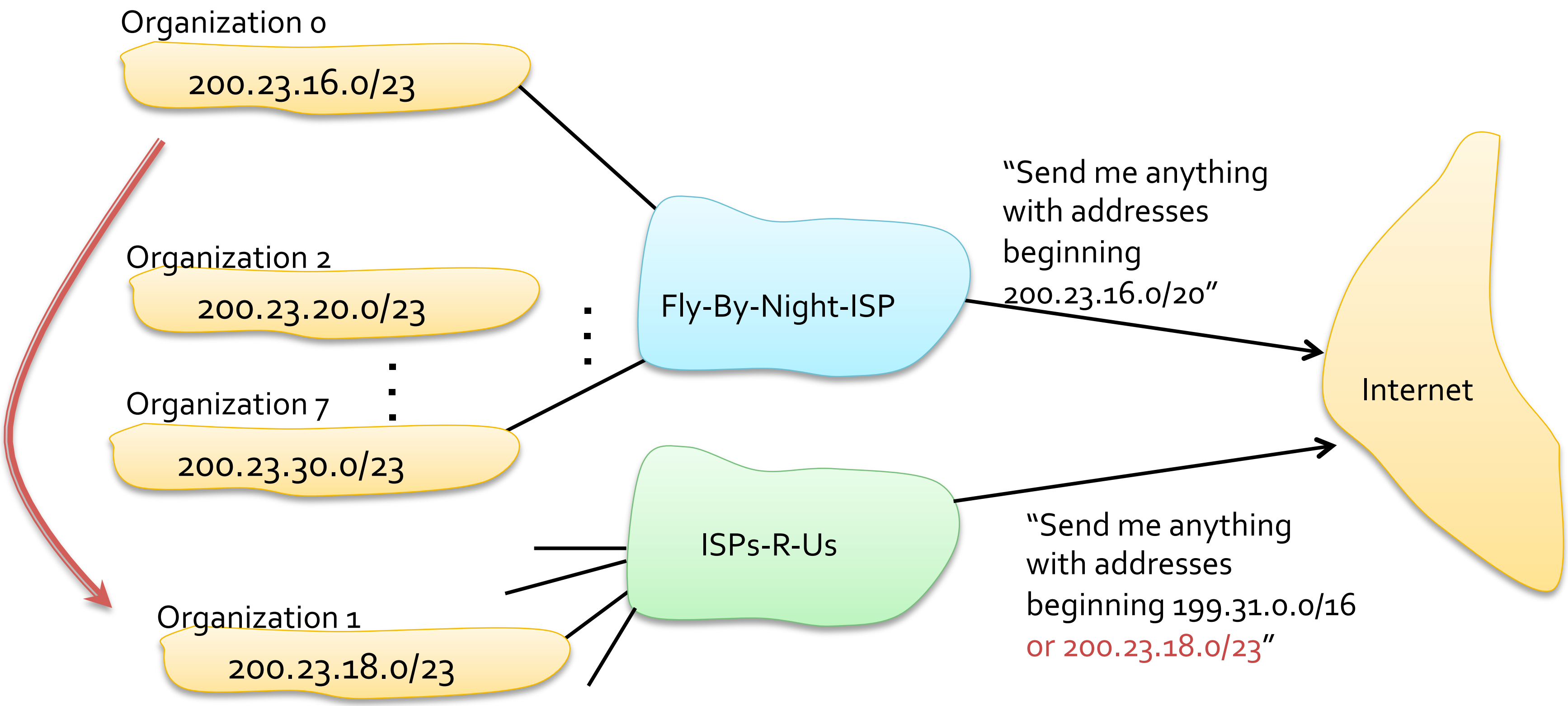
Hierarchical Addressing: Route Aggregation

Hierarchical addressing allows efficient advertisement of routing information:



Hierarchical Addressing: More Specific Routes

Let's say Organization 1 switches ISPs. What happens?
ISPs-R-Us announces a more specific route to Organization 1



How Does an ISP Get IP Addresses?

- ICANN: **I**nternet **C**orporation for **A**ssigned **N**ames and **N**umbers
  - Used to be the US Government!
  - Now a non-profit corporation
- Role of ICANN
  - Allocates addresses (actually gives large blocks of IPs to regional registries)
  - Manages DNS (actually delegates this job too)
  - Assigns domain names, resolves disputes